REMARKS

Claims 1, 3-10, 14, and 16-21 remain in the application.

The present invention relates from the discovery that by taking processing elements and forming them into one or more groups such that one group receives one instruction so that the instructions are executed in parallel, the utilization efficiency of hardware resources is improved and the time required for multimedia data processing are reduced, even when a memory stores pieces of data that requires different types of operation. Parallel processors are an extremely crowded field and consumers demand the latest features when determining which electronic equipment to utilize and/or purchase. Thus, any small improvements, no matter how minor, may be the crucial difference between commercial success or failure.

> "Thus when differences that may appear technologically minor nonetheless have a practical impact, particularly in a crowded field, the decision-maker must consider the obviousness of the new structure in this light."

Continental Can Co. USA Inc. v. Monsanto Co., 20 U.S.P.Q. 2d. 1746, 1752 (Fed. Cir. 1991).

The present invention improves the utilization efficiency of hardware resources for a computer and reduces the time required for multimedia data processing even when a memory stores pieces of data that requires different types of operations. (Spec. pg. 2, ln. 22 – pg. 3, ln. 4) The parallel execution processor 100 forms the processing elements 130a-d into one or more groups and assigns the instructions to the groups so that different groups execute different instructions in parallel or if there is only one instruction, have all the processing elements execute the one instruction. (Spec. pg. 15, ln. 8 – pg. 16, ln. 3; Fig. 1). For example, if there is only one instruction, the parallel execution processor assigns the one instruction to all of the processing elements 130a-d. However, if there are two instructions, the parallel execution

processor forms the processing elements into two groups and assigns one instruction to each of the groups. (Spec. pg. 21, lns. 4-11; Fig. 1). For example, processing elements 130a and 130c could be one group, while processing elements 130b and 130d could be another group. Processing elements 130a and 130c would then execute a first instruction while processing elements 130b and 130d would execute a second instruction in parallel. (Spec. pg. 29, lns. 2 – 24; Fig. 1).

The Office Action rejected Claims 1, 3-10, 14, and 16-21 under 35 U.S.C. §102(b) as being anticipated by *Eickemeyer et al.* (U.S. 5,355,460, hereinafter "*Eickemeyer*"). Applicant has amended the Claims to incorporate features suggested by the Examiner.

"[T]he dispositive question regarding anticipation is whether one skilled in the art would reasonably understand or infer from the prior art reference's teaching that every claim [limitation] was disclosed in that single reference." *Dayco Prods., Inc.* v. *Total Containment, Inc.*, F.3d 1358, 1368 (Fed. Cir. 2003).

Eickemeyer does not teach or suggest "a group forming unit operable to form the processing elements into as many groups as the number of instructions included in the instruction sequence." *Eickemeyer* does not form the processing elements into as many groups as the number of instructions in the instruction sequence. *Eickenmeyr* does not form processing elements into groups at all. *Eickemeyer* merely determines whether an instruction can be executed in parallel or not. If the instruction can be executed in parallel, the instruction is tagged with a "0" bit and the instruction is executed after the instruction tagged with a "1" bit.

Furthermore, the Office Action cites to Column 9, lns. 44-49 in Eickemeyer for the proposition that it forms groups of processing elements. However, *Eickemeyer* states that "In the most perfect case, where plural instructions are always being processed in parallel, the

instruction execution rate of the computer system would be N times as great as for the case where instructions are executed one at a time, with N being the number of instructions in the groups which are being processed in parallel." In *Eickemeyer*, the term "groups" refers to instructions as seen in Claim 1 of *Eickemeyer* which recites "an input/output interface for providing a group of instructions to be processed." Thus, *Eickemeyer* does not form the processing elements into groups, but merely rearranges instructions into groups such that they can be processed in parallel.

In addition, in order to realize N-parallel execution of instructions, *Eickemeyer* would need to fetch N instructions. However, in the present invention, to realize N-parallel instructions, the present invention does not need to fetch N-instructions. For example, if there was only one ADD instruction, each of the N processing elements individually executes the same ADD instruction. If, however, there are two instructions, ADD and SUB, for example, the processing element are divided into two groups A and B. All the processing elements in group A would process the ADD instruction while all of the processing elements in group B would process the SUB instruction. Thus, the present invention can realize N-parallel execution by fetching two instructions. Therefore, the present invention can realize N-parallel execution without fetching N instructions.

Eickemeyer also fails to recite "an execution controlling unit operable to assign the one or more instructions decoded by the decoding unit to the groups of the processing elements, so that each group of the processing elements receives a different one of the one or more instructions, and control the processing elements so that (i) the instructions received by the groups of the processing elements are executed in parallel, and (ii) in each group, all processing elements in the group each execute the same instruction received by the group." *Eickemeyer* also does not

disclose the execution controlling unit of the present invention. In *Eickenemeyer*, the instruction compounding unit 20 tags an instructions with a "1" bit if the instruction is a first instruction or a "0" bit if the instruction is a "second" instruction that may be executed in parallel with the first instruction. (Col. 10, lns. 19 – 27). However, *Eickemeyer* does not teach forming the processing elements in groups and thus the instructions are not assigned to groups. Instead, decoders 40, 41, and 45 determine whether instructions can be processed in parallel. Thus, decoders 40, 41, and 45 determine whether the instructions are in category A or B. Then, they determine based on the rules recited in Colum 13, lns. 25 – 29 whether the instructions can be processed in parallel or not. If the instructions can be processed in parallel, the first instruction is tagged with a "1" bit and the second instruction is tagged with a "0" bit. Otherwise, the instructions are tagged with a "1" bit only. If the following instruction is tagged with a "1" bit, then both instructions are executed singularly instead of in parallel. Thus, in Eickemeyer, the instructions are not assigned to groups of processing elements, but only bit tags to determine whether they can be processed in parallel or not.

Furthermore, there is no indication in *Eickenemeyer* that each group of processing elements receive a different instruction. In *Eickemeyer* not all of the processors are utilized since there are some instructions which are processed singularly such as when back to back instructions have a "1" bit. There is also no teaching in *Eickemeyer* that in each group of processing elements, all of the processing elements execute the same instruction received by the group since *Eickemeyer* does not break the processing elements into groups and therefore each of the processing elements in the non-existent group cannot process the same instruction.

The Office Action also states that it does not believe the limitation of multiple processing elements does not appear to be present in Claim 1. Applicant has amended Claim 1 to better clarify the limitation.

With respect to Claim 3, *Eickemeyer* fails to receite "when the number of instructions included in the instruction sequence is one, the group forming unit forms all of the processing elements into one group." The Office Action cited to "N = 1" for the proposition that the group forming unit forms all of the processing elements in one group. However, "N" represents the number of instructions in the group which are being processed in parallel and furthermore represents the speed of the computer system and not the number of processors used. If N = 1, then it only means that there is one instruction being processed at a time. That is if, there are two processors, one processor would sit idle in *Eickemeyer*. In the present invention, however, if there is only one instruction, all of the processing elements would process the single instruction and no processing element would sit idle. Thus, the structure in *Eickemeyer* is different from the structure of the present invention.

Claim 14 is a method claim directed towards Claim 1 and is patentable for at least the reasons given for Claim 1.

All arguments for patentability with respect to Claim 1 are repeated and incorporated herein for Claim 16.

Claims 3-10 and 17-21 depend from and further limit Claims 1 and 16, and are also patentable, too.

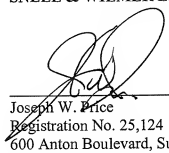
Conclusion

It is believed that the application is in condition for allowance and an early notification is requested.

If the Examiner believes a telephone interview will assist in the prosecution of this application, the undersigned attorney can be contacted at the listed phone number.

Very truly yours,

SNELL & WILMER L.L.P.



Joseph W. Price
Registration No. 25,124
600 Anton Boulevard, Suite 1400
Costa Mesa, California 92626-7689
Telephone: (714) 427-7420
Facsimile : (714) 427-7799